# Learning Goals

- State the 3-DIMENSIONAL MATCHING problem
- Understand the reduction from 3-SAT to 3-DIMENSIONAL MATCHING
- Define graph coloring and the chromatic number of a graph
- State the 3-COLORING problem

# 3-Dimensional Matching

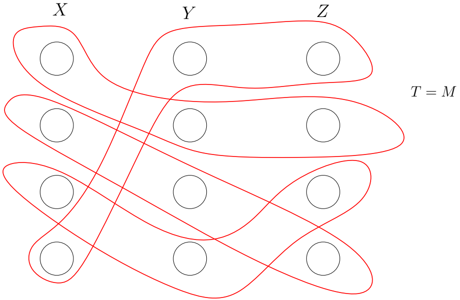
## Definition

Given disjoint sets $X$, $Y$ and $Z$, each of size $n$, and given a set $T \subseteq X \times Y \times Z$ of ordered triples, a *3-dimensional matching* is a set $M$ of $n$ triples in $T$ so that each element of $X \cup Y \cup Z$ is contained in exactly one triple in $M$.

# 3-Dimensional Matching

## Definition

Given disjoint sets $X$, $Y$ and $Z$, each of size $n$, and given a set $T \subseteq X \times Y \times Z$ of ordered triples, a *3-dimensional matching* is a set $M$ of $n$ triples in $T$ so that each element of $X \cup Y \cup Z$ is contained in exactly one triple in $M$.



# 3-DIMENSIONAL MATCHING Problem

## Definition

In the *3-DIMENSIONAL MATCHING* problem, we are given $X, Y, Z$ and $T \subseteq X \times Y \times Z$, and must decide whether there exists a 3-dimensional matching in $T$.

# 3-DIMENSIONAL MATCHING Problem

## Definition

In the *3-DIMENSIONAL MATCHING* problem, we are given $X, Y, Z$ and $T \subseteq X \times Y \times Z$, and must decide whether there exists a 3-dimensional matching in $T$.

## Theorem

*3-DIMENSIONAL MATCHING is* NP*-complete.*

# 3-DIMENSIONAL MATCHING Problem

## Definition

In the *3-DIMENSIONAL MATCHING* problem, we are given $X$, $Y$, $Z$ and $T \subseteq X \times Y \times Z$, and must decide whether there exists a 3-dimensional matching in $T$.

## Theorem

*3-DIMENSIONAL MATCHING is* NP*-complete.*

## Proof.

3-DIMENSIONAL MATCHING $\in$ NP:

- certificate: $M \subseteq T$;
- verifier: check if $M$ is a 3-dimensional matching.

# 3-DIMENSIONAL MATCHING Problem

## Definition

In the *3-DIMENSIONAL MATCHING* problem, we are given $X, Y, Z$ and $T \subseteq X \times Y \times Z$, and must decide whether there exists a 3-dimensional matching in $T$.

## Theorem

*3-DIMENSIONAL MATCHING is* NP*-complete.*

## Proof.

3-DIMENSIONAL MATCHING $\in$ NP:

- certificate: $M \subseteq T$;
- verifier: check if $M$ is a 3-dimensional matching.

We will show 3-SAT $\leq_P$ 3-DIMENSIONAL MATCHING.

# Proof Continued

**Proposition**

3-SAT $\leq_P$ 3-DIMENSIONAL MATCHING.

# Proof Continued

**Proposition**

3-SAT $\leq_P$ 3-DIMENSIONAL MATCHING.

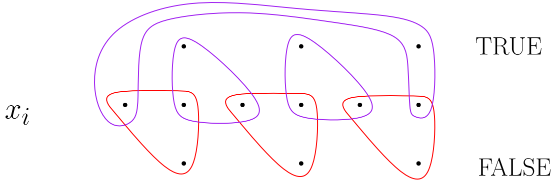
- Gadget representing TRUE and FALSE: evenness and oddity

# Proof Continued

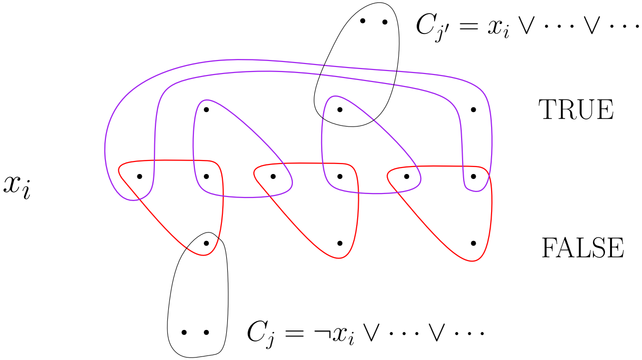
**Proposition**

3-SAT $\leq_P$ 3-DIMENSIONAL MATCHING.

- Gadget representing TRUE and FALSE: evenness and oddity

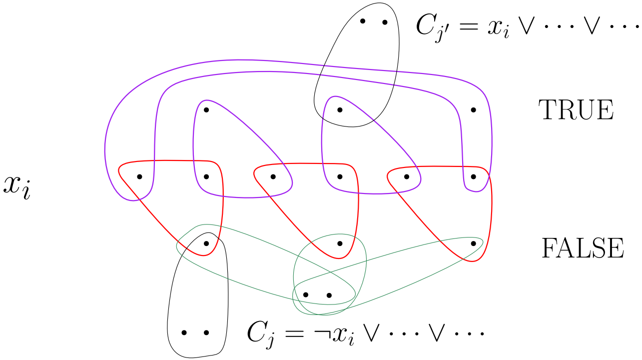


# Incorporating Clauses




Add two nodes for each clause $j$. If $x_i$ is in the clause, add a triple including the two clause nodes and the $j^{\text{th}}$ "TRUE node" of $x_i$; if $\neg x_i$ is in the clause, add a triple including the two clause nodes and the $j^{\text{th}}$ "FALSE node" of $x_i$.

# Adding cleanup nodes




Each clean up gadget consists of two cleanup nodes; add a triple for each TRUE or FALSE node of each $x_i$. How many of these are needed? To be calculated.

# Putting things together

Given a 3-SAT formula with $m$ clauses, we construct a 3-DIMENSIONAL MATCHING instance:

- Node set: (we first describe all the nodes and partition them into $X, Y, Z$ later)
  - For each variable $x_i$, create nodes $a_{i1}, \ldots, a_{i,2m}$, $T_{i,1}, \cdots, T_{i,m}$ and $F_{i,1}, \cdots, F_{i,m}$;
  - For each clause $j$, create nodes $c_{j1}$ and $c_{j2}$;
  - Create a number of cleanup gadgets; the $k^{\text{th}}$ cleanup gadget has two nodes $q_{k1}$ and $q_{k2}$.

# Putting things together

Given a 3-SAT formula with $m$ clauses, we construct a 3-DIMENSIONAL MATCHING instance:

- Node set: (we first describe all the nodes and partition them into $X, Y, Z$ later)
  - For each variable $x_i$, create nodes $a_{i1}, \ldots, a_{i,2m}$, $T_{i,1}, \cdots, T_{i,m}$ and $F_{i,1}, \cdots, F_{i,m}$;
  - For each clause $j$, create nodes $c_{j1}$ and $c_{j2}$;
  - Create a number of cleanup gadgets; the $k^{\text{th}}$ cleanup gadget has two nodes $q_{k1}$ and $q_{k2}$.
- Triple set $T$:
  - For each $x_i$, for $j = 1, \cdots, m$, create triples $(a_{i,2j-1}, a_{i,2j}, F_{i,j})$ and $(a_{i,2j}, a_{i,2j+1}, T_{i,j})$ (interpreting $a_{i,2m+1}$ as $a_{i,1}$)
  - For each clause $j$, for each positive literal $x_i$ in it, add triple $(c_{j1}, c_{j2}, T_{ij})$; for each negative literal $\neg x_i$ in the clause, add triple $(c_{j1}, c_{j2}, F_{ij})$;
  - For each cleanup gadget $k$, for each $x_i$ and clause $j$, add $(q_{k1}, q_{k2}, T_{ij})$ and $(q_{k1}, q_{k2}, F_{ij})$.

# The last step

- How many clean up gadgets do we need?

# The last step

- How many clean up gadgets do we need?
  - We have $n$ variables, each having $m$ TRUE nodes and $m$ FALSE nodes; half of these are "free"; each clause uses one such node, leaving $mn - m$ "free" nodes to be cleaned up.

# The last step

- How many clean up gadgets do we need?
  - We have $n$ variables, each having $m$ TRUE nodes and $m$ FALSE nodes; half of these are "free"; each clause uses one such node, leaving $mn - m$ "free" nodes to be cleaned up.
  - Therefore we need $m(n-1)$ cleanup gadgets.

# The last step

- How many clean up gadgets do we need?
  - We have $n$ variables, each having $m$ TRUE nodes and $m$ FALSE nodes; half of these are "free"; each clause uses one such node, leaving $mn - m$ "free" nodes to be cleaned up.
  - Therefore we need $m(n-1)$ cleanup gadgets.
- Partition the nodes into $X, Y, Z$.
  - All the odd numbered "$a, c, q$" nodes form $X$;
  - All the even numbered "$a, c, q$" nodes form $Y$;
  - All the TRUE and FALSE nodes form $Z$.

# The last step

- How many clean up gadgets do we need?
  - We have $n$ variables, each having $m$ TRUE nodes and $m$ FALSE nodes; half of these are "free"; each clause uses one such node, leaving $mn - m$ "free" nodes to be cleaned up.
  - Therefore we need $m(n-1)$ cleanup gadgets.
- Partition the nodes into $X, Y, Z$.
  - All the odd numbered "$a, c, q$" nodes form $X$;
  - All the even numbered "$a, c, q$" nodes form $Y$;
  - All the TRUE and FALSE nodes form $Z$.

**Claim**

The 3-SAT formula is satisfiable if and only if the instance we constructed has a 3-dimensional matching.

# Graph coloring

**Definition**

A *coloring* of an undirected graph is an assignment of colors to vertices, so that no two adjacent vertices share the same color.

# Graph coloring

## Definition

A *coloring* of an undirected graph is an assignment of colors to vertices, so that no two adjacent vertices share the same color.

In graph theory, the minimal number of colors for which such an assignment is possible is called the *chromatic number*, often denoted as $\chi(G)$ for graph $G$.

# Graph coloring

## Definition

A *coloring* of an undirected graph is an assignment of colors to vertices, so that no two adjacent vertices share the same color.

In graph theory, the minimal number of colors for which such an assignment is possible is called the *chromatic number*, often denoted as $\chi(G)$ for graph $G$.

## Definition

For the prolbem *3-COLORING*, we are given a graph $G$, and must decide whether $\chi(G) \leq 3$, i.e., whether $G$ can be colored using three colors.

# Graph coloring

**Definition**

A *coloring* of an undirected graph is an assignment of colors to vertices, so that no two adjacent vertices share the same color.

In graph theory, the minimal number of colors for which such an assignment is possible is called the *chromatic number*, often denoted as $\chi(G)$ for graph $G$.

**Definition**

For the prolbem *3-COLORING*, we are given a graph $G$, and must decide whether $\chi(G) \leq 3$, i.e., whether $G$ can be colored using three colors.

**Proposition**

$\chi(G) \leq 2$ if and only if $G$ is bipartite, and this can be checked in linear time.

# 3-COLORING Problem

**Theorem**

*3-COLORING is* NP-*complete.*

Proof omitted. (Optional reading.)

# 3-COLORING Problem

**Theorem**

*3-COLORING is* NP*-complete.*

Proof omitted. (Optional reading.)

**Definition**

For a (simple) graph $G = (V, E)$, its *complement graph* $\overline{G}$ is has the same node set $V$, and any two distinct vertices $u$ and $v$ are adjacent in $\overline{G}$ if and only if they are not adjacent in $G$.

# 3-COLORING Problem

**Theorem**

*3-COLORING is* NP*-complete.*

Proof omitted. (Optional reading.)

**Definition**

For a (simple) graph $G = (V, E)$, its *complement graph* $\overline{G}$ is has the same node set $V$, and any two distinct vertices $u$ and $v$ are adjacent in $\overline{G}$ if and only if they are not adjacent in $G$.

In graph theory, the size of a largest independent set of a graph $G$ is called its *independence number*, often denoted as $\alpha(G)$.

# 3-COLORING Problem

**Theorem**

*3-COLORING is* NP*-complete.*

Proof omitted. (Optional reading.)

**Definition**

For a (simple) graph $G = (V, E)$, its *complement graph* $\overline{G}$ is has the same node set $V$, and any two distinct vertices $u$ and $v$ are adjacent in $\overline{G}$ if and only if they are not adjacent in $G$.

In graph theory, the size of a largest independent set of a graph $G$ is called its *independence number*, often denoted as $\alpha(G)$.

**Proposition (Exercise)**

$\chi(G) \geq \alpha(\overline{G})$.

# Exercises on coloring

**Proposition (Exercise)**

$\chi(G) \geq \alpha(\overline{G})$.

**Proof.**

An independent set of size $k$ in $\overline{G}$ is a *clique* of size $k$ in $G$, i.e., every pair of these $k$ nodes are connected. $k$ colors are needed for them. □

**Proposition (Exercise)**

3-COLORING $\leq_P$ $k$-COLORING

**Proof.**

Given a graph $G$, add $k-3$ nodes to $G$, all connected with each other and all connected with each node in $G$. The new graph is $k$ colorable if and only if $G$ is 3-colorable. □

# Exercises

**Question**

Give a graph $G$ where $\chi(G) > \alpha(\overline{G})$.

**Question**

In the 3-EXACT COVER problem, we are given a universe $U$ and a list of subsets $S_1, \cdots, S_m \subseteq U$ each of cardinality 3. We must decide whether there exists a subset of these subsets that cover each element in $U$ exactly once. Show that 3-EXACT COVER is NP complete.

# Exercises

Are the following problems NP-complete or polynomial-time solvable? (Assuming NP $\neq$ P.)

1. Given a graph $G$, decide whether there is an independent set of size 1000.

# Exercises

Are the following problems NP-complete or polynomial-time solvable? (Assuming NP $\neq$ P.)

1. Given a graph $G$, decide whether there is an independent set of size 1000.
2. (Hitting Set) Given a set $A = \{a_1, \ldots, a_n\}$, a collection $B_1, B_2, \cdots, B_m$ of subsets of $A$, and an integer $k > 0$. Is there a set $H \subseteq A$, $|H| \leq k$ such that $H \cap B_i \neq \emptyset$ for $i = 1, \ldots, m$?

# Exercises

Are the following problems NP-complete or polynomial-time solvable? (Assuming NP $\neq$ P.)

1. Given a graph $G$, decide whether there is an independent set of size 1000.
2. (Hitting Set) Given a set $A = \{a_1, \ldots, a_n\}$, a collection $B_1, B_2, \cdots, B_m$ of subsets of $A$, and an integer $k > 0$. Is there a set $H \subseteq A$, $|H| \leq k$ such that $H \cap B_i \neq \emptyset$ for $i = 1, \ldots, m$?
3. Given a directed graph $G = (V, E)$ with $s, t \in V$, and an integer $k > 0$.
   1. Does there exist at least $k$ edge-disjoint paths from $s$ to $t$?

# Exercises

Are the following problems NP-complete or polynomial-time solvable? (Assuming NP $\neq$ P.)

1. Given a graph $G$, decide whether there is an independent set of size 1000.
2. (Hitting Set) Given a set $A = \{a_1, \ldots, a_n\}$, a collection $B_1, B_2, \cdots, B_m$ of subsets of $A$, and an integer $k > 0$. Is there a set $H \subseteq A$, $|H| \leq k$ such that $H \cap B_i \neq \emptyset$ for $i = 1, \ldots, m$?
3. Given a directed graph $G = (V, E)$ with $s, t \in V$, and an integer $k > 0$.
    1. Does there exist at least $k$ edge-disjoint paths from $s$ to $t$?
    2. Given $m$ paths $P_1, \cdots, P_m$ from $s$ to $t$, does there exist at least $k$ paths among $P_1, \cdots, P_m$ that are edge-disjoint?

# Exercises

Are the following problems NP-complete or polynomial-time solvable? (Assuming NP $\neq$ P.)

1. Given an undirected graph and integer $k$, decide whether there is a spanning tree in which each node has degree at most $k$.

# Exercises

Are the following problems NP-complete or polynomial-time solvable? (Assuming NP $\neq$ P.)

1. Given an undirected graph and integer $k$, decide whether there is a spanning tree in which each node has degree at most $k$.
2. Given a directed graph $G = (V, E)$ with $s, t \in V$ and nonnegative integral edge weights, and an integer $k > 0$.
    1. Does there exist a simple path from $s$ to $t$ with total weight at most $k$?

# Exercises

Are the following problems NP-complete or polynomial-time solvable? (Assuming NP $\neq$ P.)

1. Given an undirected graph and integer $k$, decide whether there is a spanning tree in which each node has degree at most $k$.
2. Given a directed graph $G = (V, E)$ with $s, t \in V$ and nonnegative integral edge weights, and an integer $k > 0$.
    1. Does there exist a simple path from $s$ to $t$ with total weight at most $k$?
    2. Does there exist a simple path from $s$ to $t$ with total weight at least $k$?